The Digital Abstraction
Review

- **Discretize matter** by agreeing to observe the lumped matter discipline

  ![Lumped Circuit Abstraction](image)

  Lumped Circuit Abstraction

- **Analysis tool kit**: KVL/KCL, node method, superposition, Thévenin, Norton (remember superposition, Thévenin, Norton apply only for linear circuits)
Today

Discretize value $\rightarrow$ Digital abstraction

Interestingly, we will see shortly that the tools learned in the previous three lectures are sufficient to analyze simple digital circuits

Reading: Chapter 5 of Agarwal & Lang
But first, why digital?
In the past …

Analog signal processing

\[
V_0 = \frac{R_2}{R_1 + R_2} V_1 + \frac{R_1}{R_1 + R_2} V_2
\]

If \( R_1 = R_2 \),

\[
V_0 = \frac{V_1 + V_2}{2}
\]

The above is an “adder” circuit.
Noise Problem

... noise hampers our ability to distinguish between small differences in value — e.g. between 3.1V and 3.2V.
Value Discretization

Restrict values to be one of two

\[\begin{array}{cccc}
\text{HIGH} & \text{LOW} \\
5V & 0V \\
TRUE & FALSE \\
1 & 0 \\
\end{array}\]

...like two digits 0 and 1

Why is this discretization useful?

(Remember, numbers larger than 1 can be represented using multiple binary digits and coding, much like using multiple decimal digits to represent numbers greater than 9. E.g., the binary number 101 has decimal value 5.)
Digital System

sender

\[ V_N = 0V \]

receiver

\[ V_N = 0.2V \]

With noise

Cite as: Anant Agarwal and Jeffrey Lang, course materials for 6.002 Circuits and Electronics, Spring 2007. MIT OpenCourseWare (http://ocw.mit.edu/), Massachusetts Institute of Technology. Downloaded on [DD Month YYYY].
Digital System

Better noise immunity
Lots of “noise margin”

For “1”: noise margin \(5V \text{ to } 2.5V = 2.5V\)
For “0”: noise margin \(0V \text{ to } 2.5V = 2.5V\)
Voltage Thresholds and Logic Values

1

sender

0

1

1

receiver

2.5V

0V

5V

0V

2.5V

0

1
But, but, but ...  
What about 2.5V?

Hmmm... create “no man’s land” or forbidden region

For example,

```
V_H
V_L
```

```
0V
2V
3V
5V
```

```
0  1
```

```
0V  2V  3V  5V
```

```
V_H
V_L
```

```
V_H  5V
```

```
V_L  0V
```

But, but, but ...  
What about 2.5V?

Hmmm... create “no man’s land” or forbidden region

For example,
But, but, but …
Where’s the noise margin?
What if the sender sent 1: $V_H$?

Hold the sender to tougher standards!
But, but, but ...  
Where's the noise margin?  
What if the sender sent 1: $V_H$?

Hold the sender to tougher standards!

"1" noise margin: $V_{IH} - V_{OH}$  
"0" noise margin: $V_{IL} - V_{OL}$
Digital systems follow static discipline: if inputs to the digital system meet valid input thresholds, then the system guarantees its outputs will meet valid output thresholds.
Processing digital signals

Recall, we have only two values —

\[ 1, 0 \iff \text{Map naturally to logic: } T, F \]
\[ \iff \text{Can also represent numbers} \]
Processing digital signals

Boolean Logic

If $X$ is true and $Y$ is true
Then $Z$ is true else $Z$ is false.

$Z = \overline{X} \cdot \overline{Y}$

$Z = X \cdot Y$

$X, Y, Z$ are digital signals
“0”, “1”

Truth table representation:

<table>
<thead>
<tr>
<th>$X$</th>
<th>$Y$</th>
<th>$Z$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Enumerate all input combinations
Combinational gate abstraction

- Adheres to static discipline
- Outputs are a function of inputs alone.

Digital logic designers do not have to care about what is inside a gate.
Z = X \cdot Y
Examples for recitation

\[ Z = X \cdot Y \]
In recitation...

Another example of a gate

If \((A \text{ is true}) \text{ OR } (B \text{ is true})\)
then \(C\) is true
else \(C\) is false

\[ C = A + B \quad \text{Boolean equation} \]

\[ \Rightarrow \quad A \quad \text{OR gate} \quad B \quad C \]

More gates

\[ \quad B \quad \text{Inverter} \quad \bar{B} \]

\[ X \quad Y \quad \text{NAND} \quad Z \]

\[ Z = \overline{X \cdot Y} \]
**Boolean Identities**

\[
\begin{align*}
X \cdot 1 &= X \\
X \cdot 0 &= X \\
X + 1 &= 1 \\
X + 0 &= X \\
\overline{1} &= 0 \\
\overline{0} &= 1 \\
AB + AC &= A \cdot (B + C)
\end{align*}
\]

**Digital Circuits**

Implement: \( \text{output} = A + B \cdot C \)

![Circuit Diagram](image)

Cite as: Anant Agarwal and Jeffrey Lang, course materials for 6.002 Circuits and Electronics, Spring 2007. MIT OpenCourseWare (http://ocw.mit.edu/), Massachusetts Institute of Technology. Downloaded on [DD Month YYYY].